

# NODE.JS

Michael Gfeller

# Inhaltsverzeichnis

- Einleitung / Motivation
- Konzepte
- Module
- API

Code-Beispiele auf Github: <https://github.com/gfeller/NodeJS>

## Die Teilnehmer...

- verstehen den unterschied zwischen Server und Client JavaScript Code.
- kennen den Unterschied zwischen klassischen und event-driven Web Services.
- können die Konzepte von Node.js anwenden.
- können mit dem Online API von Node.js umgehen.
- können mit Node.js einen rudimentären Server implementieren.

# **EINLEITUNG**

# Weshalb schauen wir den Web-Server an?

- **Über den Horizont schauen**
- **Bei den fürs Front-End relevanten Server-Schnittstellen mitreden können**
- **Ideal zum erstellen eines Front-End Prototypen**

**Was muss ein Web Server können?**

**Was sollte ein Web Server können?**

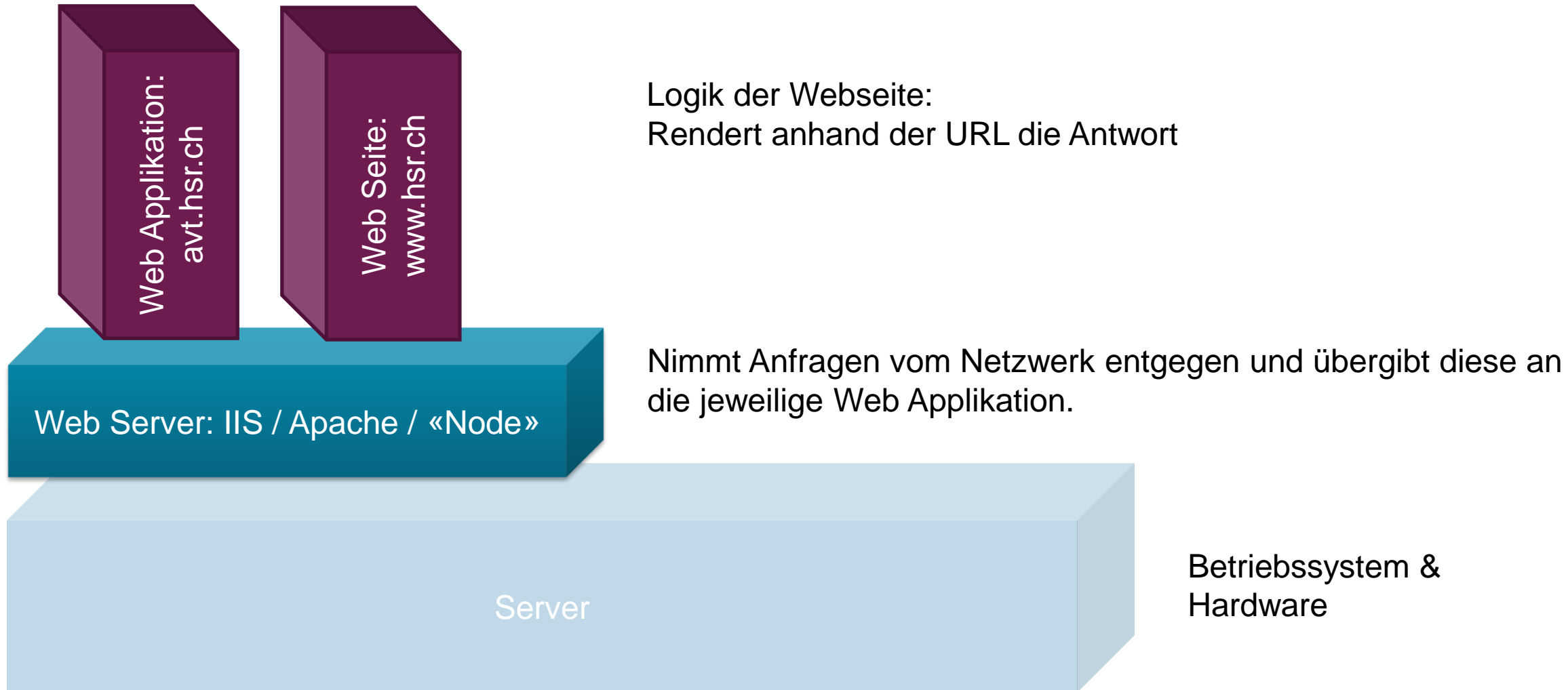
- **HTTP Anfragen annehmen**
- **Actions ausführen basierend auf der Anfrage URL**
- **HTTP Antworten absenden**
- **...**

**Beispiele:**

- **Statische Inhalte ausliefern ( HTML, CSS, JavaScript)**
- **Dynamische Inhalte rendern ( HTML erstellen aus dynamischen Daten)**
- **Anbinden von Datenbanken**



# Web Server und Web Applikation oder Web Seite



- **JavaScript**
  - Client und Server in «gleicher Sprache»
- **Ideal für Datenschnittstellen wie REST**
  - Nativer Support für JSON
- **Läuft überall**
- **Einfach zu deployen**
- **Sehr schnelle Entwicklung möglich**
- **Sehr modular Aufbau**
- **Wenig «Magie»**



# Was ist Node.js?

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#). Node.js uses an **event-driven, non-blocking I/O model** that makes it lightweight and efficient. **Node.js' package ecosystem**, [npm](#), is the largest ecosystem of open source libraries in the world.

(Quelle: <https://nodejs.org/>)

## ■ Event-based, asynchronous I/O

- Basiert auf dem Reactor Pattern: [http://en.wikipedia.org/wiki/Reactor\\_pattern](http://en.wikipedia.org/wiki/Reactor_pattern)
  - Single Threaded

## ■ Callbacks, Events, Streams and Modules

## ■ Clean, consistent API

- z.B. Fehler Behandlung
- Kein DOM, keine Browser API

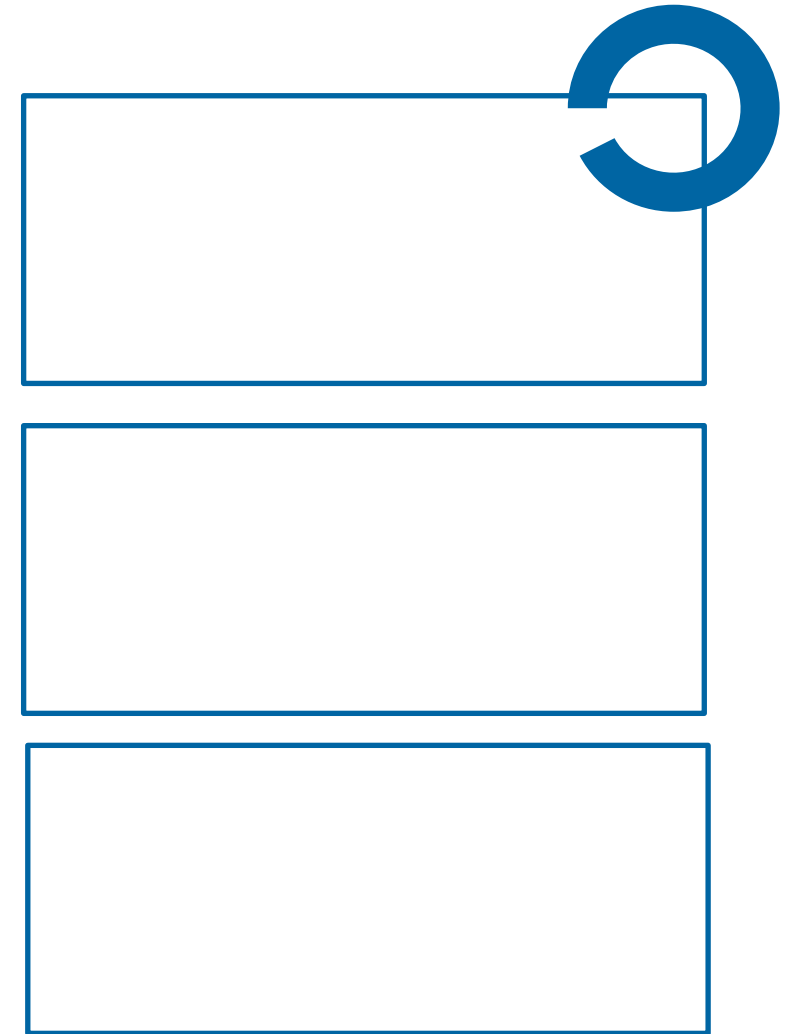
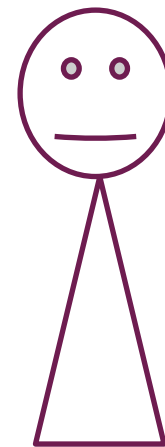
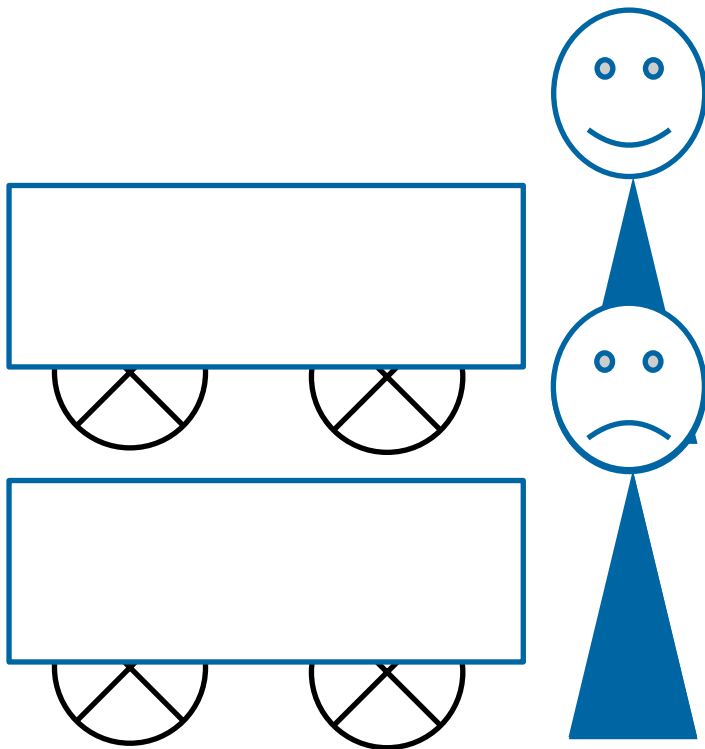
# Was ist Node.js nicht

- **Web-Framework wie Ruby on Rails, ASP.NET, Zend Framework**
  - Kann aber dazu erweitert werden
- **Programmiersprache**

**EVENT-DRIVEN, NON-BLOCKING**

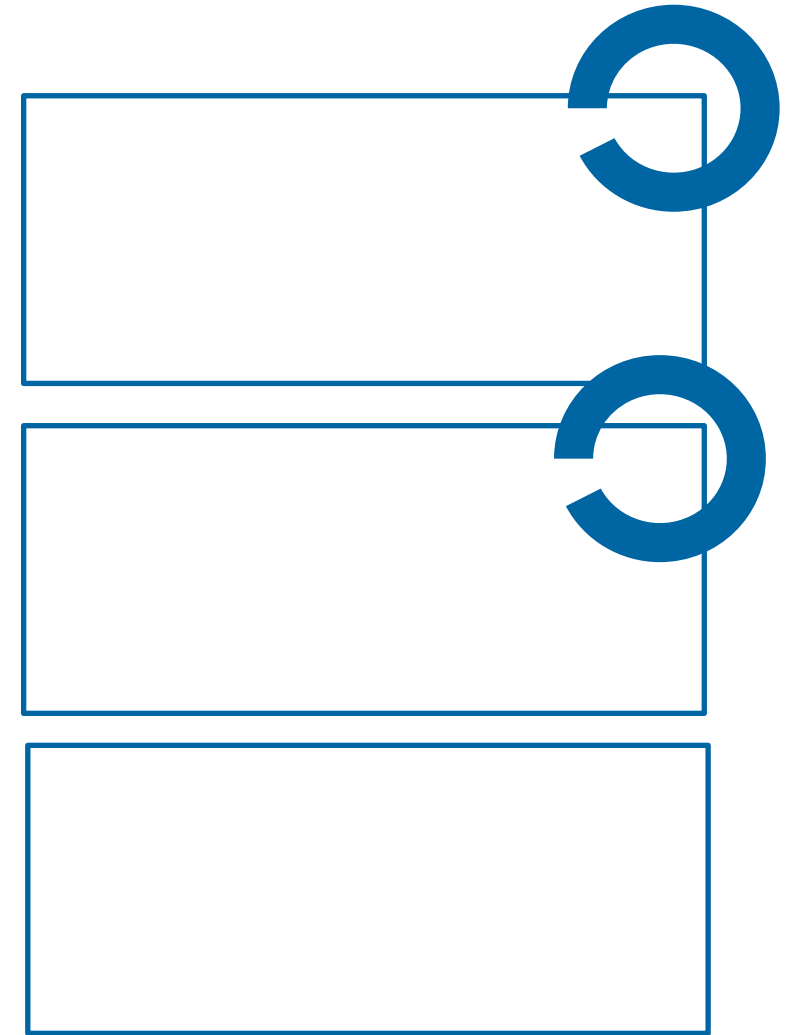
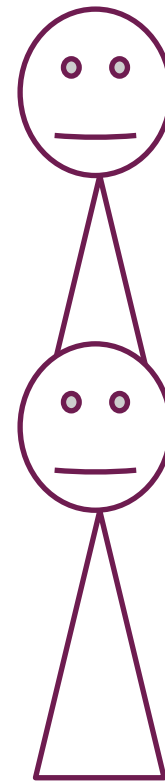
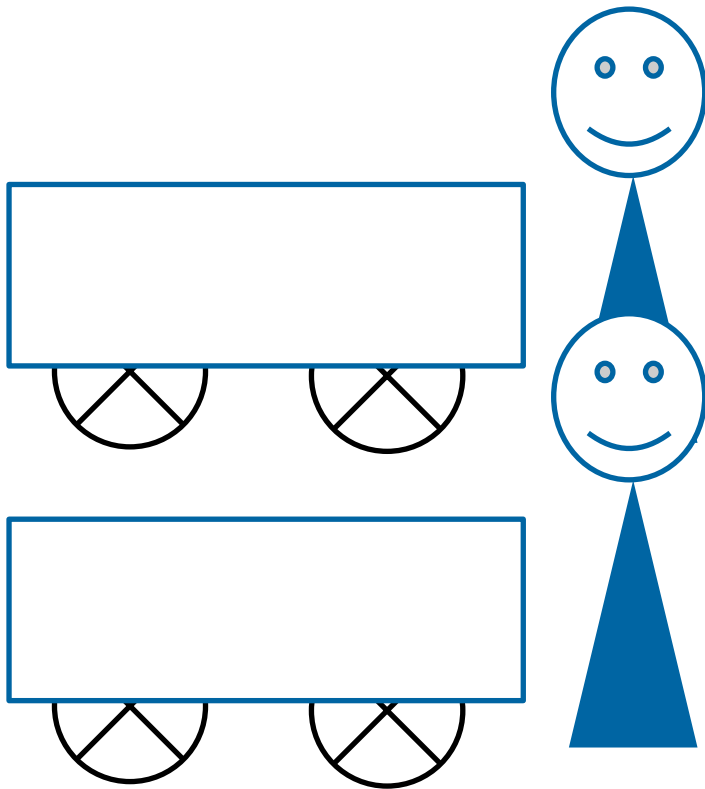
# Non-Blocking - Event-Driven!

Mehrere Clients : 1 Server Prozesse



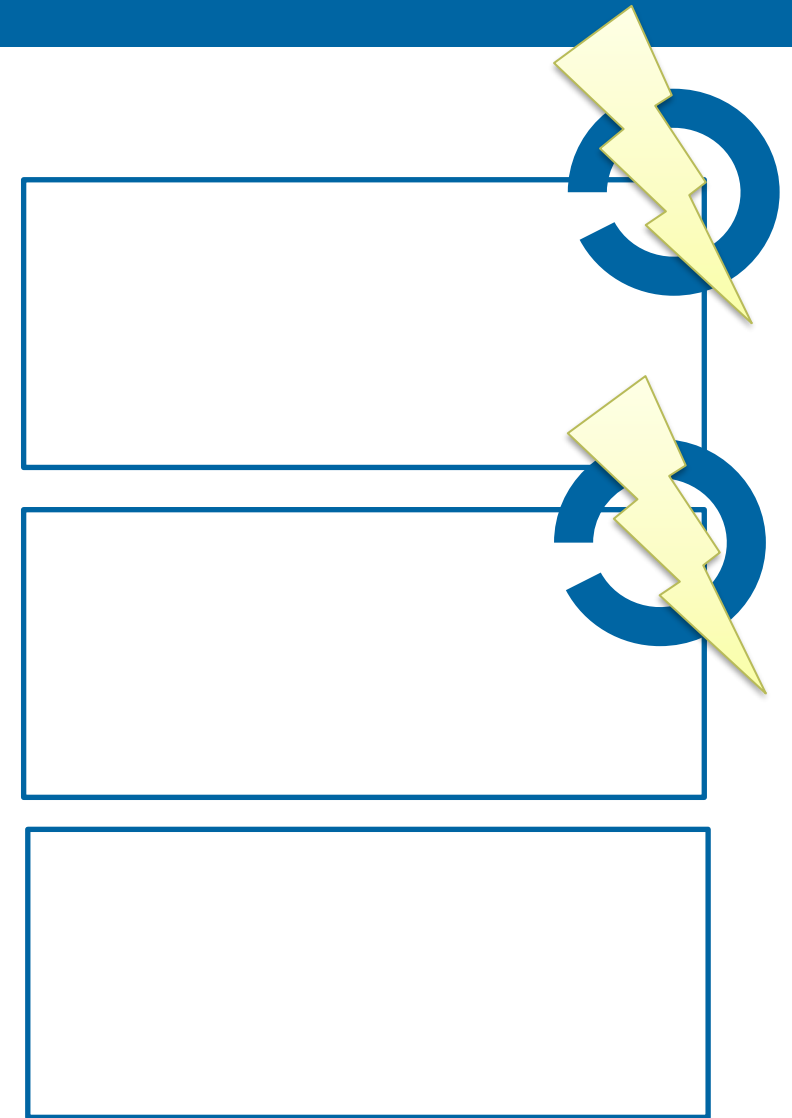
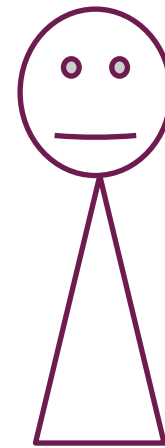
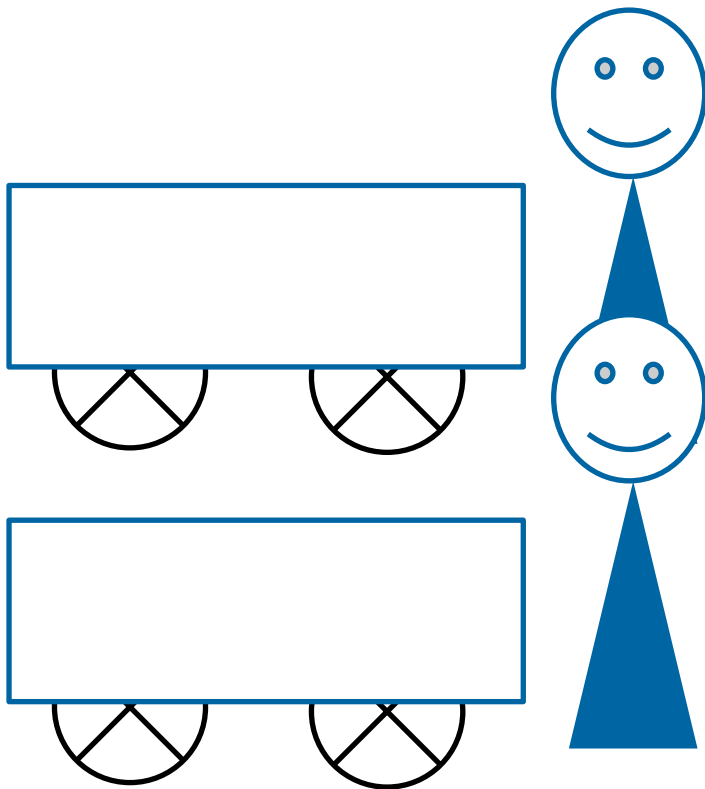
# Non-Blocking - Event-Driven!

Mehrere Clients : mehrere Server Prozesse



# Non-Blocking - Event-Driven!

Mehrere Clients : 1 Server Prozess - Event-Basiert



# Non-Blocking - Event-Driven!

## ■ Welches dieser Methode ist die beste?

- Kosten
  - Strom
- Skalierung
  - Prozessoren sind limitiert.
- Aufgabe
  - Viele kleine Aufgaben
  - Wenig aber grosse Aufgaben
  - Viele und grosse Aufgaben



# NODE.JS GRUNDLAGEN

## ■ Funktionen können als Parameter übergeben werden.

```
function myFunc(a, b, fn) {  
    setTimeout(function() {  
        fn(a + b);  
    }, 1000);  
}  
  
myFunc(2, 4, console.log);  
myFunc(10, 3, console.error );
```

Diese Funktion wird später aufgerufen

## ■ Callback-Hell

- Lösung: [Promise](#)
  - Besser: [async / await](#)
- Aktuell nutzt Node.js Callbacks und nicht Promises
  - Aber: Für viele Module gibt es Module welche den Promise-Support ergänzen  
Beispiele: <https://www.npmjs.com/package/fs-extra> / <https://www.npmjs.com/package/nedb-promise>
  - Aber: [https://nodejs.org/dist/latest-v10.x/docs/api/fs.html#fs\\_fs\\_promises\\_api](https://nodejs.org/dist/latest-v10.x/docs/api/fs.html#fs_fs_promises_api)

- Event ist ein Pattern: [http://en.wikipedia.org/wiki/Observer\\_pattern](http://en.wikipedia.org/wiki/Observer_pattern)
- Callbacks sind 1 : 1 Verbindungen. Events sind 1 : \* Verbindungen

```
const button = document.getElementById("send-button");

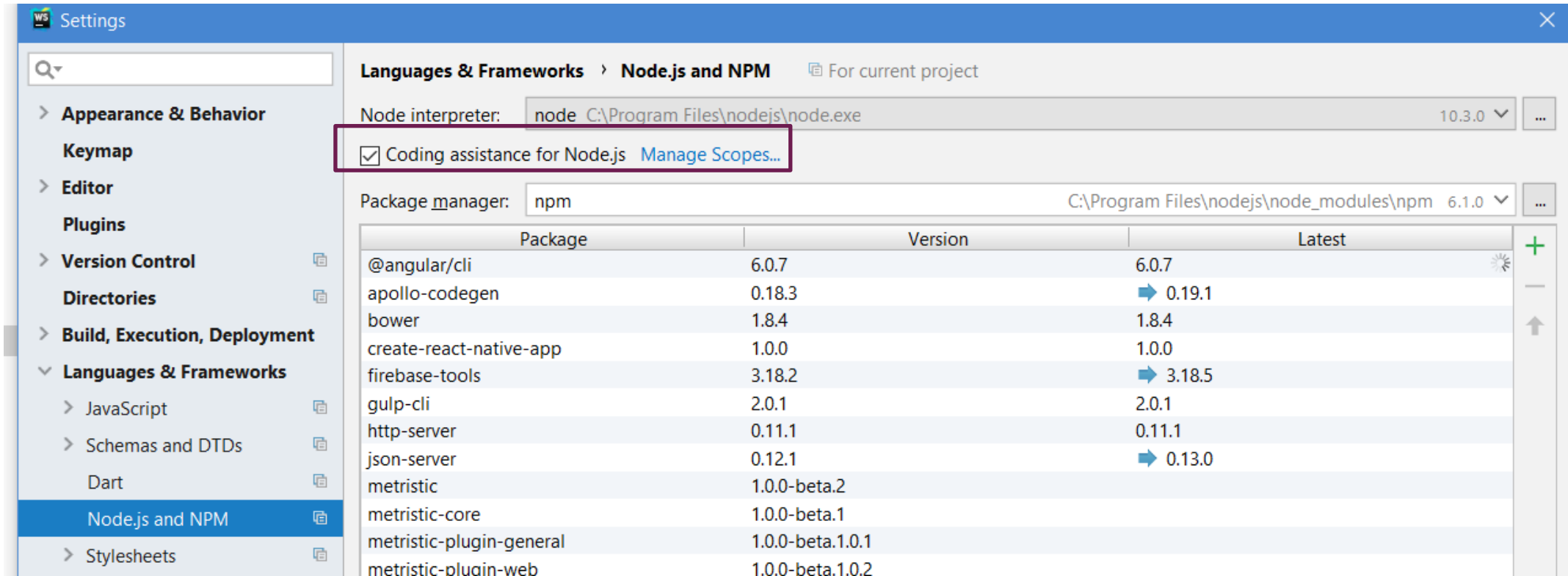
button.addEventListener('click', function (event) {
  console.log("1. subscription");
});
button.addEventListener('click', function (event) {
  console.log("2. subscription");
});
button.addEventListener('click', function (event) {
  console.log("3. subscription");
});
```

# Core Modules

## ■ <https://nodejs.org/api/>

- HTTP/HTTPS
- URL
- FS: FileSystem
- Console
- UDP / Net
- Crypto
- DNS
- ...

## ■ Node.js aktivieren



Settings

WS

Search

Appearance & Behavior

Keymap

Editor

Plugins

Version Control

Directories

Build, Execution, Deployment

▼ Languages & Frameworks

JavaScript

Schemas and DTDs

Dart

Node.js and NPM

Stylesheets

Languages & Frameworks > Node.js and NPM For current project

Node interpreter: node C:\Program Files\nodejs\node.exe 10.3.0 ...

☒ Coding assistance for Node.js [Manage Scopes...](#)

Package manager: npm C:\Program Files\nodejs\node\_modules\npm 6.1.0 ...

Package	Version	Latest
@angular/cli	6.0.7	6.0.7
apollo-codegen	0.18.3	➡ 0.19.1
bower	1.8.4	1.8.4
create-react-native-app	1.0.0	1.0.0
firebase-tools	3.18.2	➡ 3.18.5
gulp-cli	2.0.1	2.0.1
http-server	0.11.1	0.11.1
json-server	0.12.1	➡ 0.13.0
metristic	1.0.0-beta.2	
metristic-core	1.0.0-beta.1	
metristic-plugin-general	1.0.0-beta.1.0.1	
metristic-plugin-web	1.0.0-beta.1.0.2	

# ERSTES BEISPIEL

- <https://nodejs.org/>
- <https://nodejs.org/api/http.html>
- [https://nodejs.org/api/http.html#http\\_class\\_http\\_serverresponse](https://nodejs.org/api/http.html#http_class_http_serverresponse)
- [https://nodejs.org/api/http.html#http\\_class\\_http\\_clientrequest](https://nodejs.org/api/http.html#http_class_http_clientrequest)
- => DEMO



# Request (IncomingMessage)

- **method**

- GET, PUT, POST, ...

- **url**

- Angefragte url z.B. /help or help?page=1

# Response

## ■ writeHead

- `response.writeHead(200, {'Content-Length': body.length, 'Content-Type': 'text/plain' });`

## ■ setHeader

- `response.setHeader("Content-Type", "text/html");`

## ■ statusCode

- `response.statusCode = 404;`

## ■ statusMessage

- `response.statusMessage = 'Not found';`

## ■ write

- `response.write("Data")`

## ■ end

- `response.end("Data")`

**MODULE**

- **Node verwendet für die Module Verwaltung npm:** <https://www.npmjs.com/>
  - npm init
  - npm install [module name] -S -g
- **Node verwendet :** <http://www.commonjs.org/>
  - [ES6 Syntax](#) ab 8.5 möglich. ( --experimental-modules )  
<http://2ality.com/2017/09/native-esm-node.html>
    - Nächste Woche
- **Module-Files werden nur einmal geladen.**
- **Implementieren**
  - Native
  - JavaScript

## Export:

```
module.exports = myExportedObject;
```

## Beispiel:

```
let counter = 0;
function add() { return ++counter; }
function get() { return counter; }

module.exports = { count : add, get : get };
```

- Wird nur einmal durchlaufen. Das Objekt wird «ge-cached» und bei jeder Nachfrage wieder zurückgegeben.
- Objekt wird «geshared» und kann von andern Modulen theoretisch angepasst werden.

## Import:

```
const myModule = require('moduleName');  
const myModule = require('./moduleName.js');
```

## Resolve-Rheinfoolge

1. **Core Module z.B. require(«HTTP»)**
2. **Falls der mit mit «.» «..» oder «\» startet z.B. require(«.\myModule»)**
  1. Suche als File
  2. Suche als Directory
3. **Falls ein «Filename» angegeben wurde z.B. require(«myModule»)**
  1. Wird ein Module im «node\_modules» gesucht
    1. Wird gesucht bis zum «File-System-Root»

[https://nodejs.org/api/modules.html#modules\\_all\\_together](https://nodejs.org/api/modules.html#modules_all_together)

# package.json

- Beinhaltet die Informationen zum Projekt
- Wird benötigt um es zu publishen
- Wird benötigt um Module zu installieren
- Definiert Skripts
  - Ausführen npm run <script-name>
    - Beispiel npm run test

```
{  
  "name": "my_package",  
  "description": "",  
  "version": "1.0.0",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "repository": {  
    "type": "git",  
    "url": "https://github.com/mgfeller/my_package.git"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC",  
  "bugs": {  
    "url": "https://github.com/mgfeller/my_package/issues"  
  },  
  "homepage": "https://github.com/mgfeller/my_package"  
}
```

<https://docs.npmjs.com/files/package.json>



# package-lock.json

- Beschreibt den exakten Abhängigkeitsgraph vom Projekt.
- Garantiert, dass immer die gleichen Abhängigkeiten installiert werden.
- Beschleunigt die Installation.
- Wird automatisch aktualisiert wenn npm package.json oder node\_modules anpasst.
- Gehört in das Git-Repo.

```
{
  "name": "my_package",
  "version": "1.0.0",
  "lockfileVersion": 1,
  "requires": true,
  "dependencies": {
    "fancy-calc-demo": {
      "version": "5.0.2",
      "resolved": "https://registry.npmjs.org/fancy-calc-demo/-/fancy-calc-demo-5.0.2.tgz",
      "integrity": "sha512-93xBMjZMU6HfGLXlwi1uYtQKL6eiNqOsWqkuFm1WMGJqKVBMF3+jb/dSrsRHrvpp1IIDxUk1wLNUzeZ5BTMjwQ=="
    }
  }
}
```

- <https://docs.npmjs.com/configuring-npm/package-lock-json.html>

## ■ Viele Node.js Module verwenden

- `var result = fs.readFileSync()`
- `var binding = process.binding('fs').`
- `var r = binding.read(fd, buffer, offset, length, position); // sync`
- `=> Native Code!`

## ■ Beispiel für ein «Custom» native Module

- <https://github.com/ErikDubbelboer/node-sleep>
- Installation: <https://github.com/nodejs/node-gyp#installation>

## ■ Weitere Informationen:

- <https://nodejs.org/api/addons.html>
- <https://github.com/libuv/libuv>
- Neue API: <https://nodejs.org/dist/latest-v10.x/docs/api/n-api.html>

## ■ Merke: `node_modules` nicht ausliefern. Könnte Betriebssystem abhängig sein.

```
#define ASYNC_CALL(func, callback, ...)
FSReqWrap* req_wrap = new FSReqWrap(#func);
int r = uv_fs_##func(uv_default_loop(), &req_wrap->req_,
    __VA_ARGS__, After);
req_wrap->object_->Set(oncomplete_sym, callback);
req_wrap->Dispatched();
if (r < 0) {
    uv_fs_t* req = &req_wrap->req_;
    req->result = r;
    req->path = NULL;
    req->errno = uv_last_error(uv_default_loop()).code;
    After(req);
}
return scope.Close(req_wrap->object_);
```

# Zyklus

main.js:

```
let a = require("./a");  
let b = require("./b");  
a.showObject();  
b.showObject();
```

a.js:

```
let b = require("./b");  
console.log("A");  
module.exports = {showObject : function(){ console.log("A", b)}};
```

b.js:

```
let a = require("./a");  
console.log("B");  
module.exports = {showObject : function(){ console.log("B", a)}};
```

**Merke: Zyklus verhindern**

- **Problematik: Projekte benötigen unterschiedliche Version von node.js**

- **Bekannteste Lösung ist nvm**

- <https://github.com/nvm-sh/nvm>
- <https://github.com/coreybutler/nvm-windows>

```
C:\>nvm install latest
Downloading node.js version 14.3.0 (64-bit)...
Complete
Creating C:\Users\mgfeller\AppData\Roaming\nvm\temp

Downloading npm version 6.14.5... Complete
Installing npm v6.14.5...

Installation complete. If you want to use this version, type

nvm use 14.3.0

C:\>
C:\>nvm list

 14.3.0
* 12.10.0 (Currently using 64-bit executable)
 12.5.0
 12.3.1
 10.5.0
  6.3.0
```

**API**

- Node hat eine sehr umfangreiche API: <https://nodejs.org/api/>

- Typisches Beispiel:

```
let fs = require('fs');
let path = "test.txt";

fs.readFile(path, function(err, content) {
  if (err) return console.error(err);
  console.log('content of file: %s\n', path);
  console.log(content.toString());
});
```

- Der Callback ist immer das letzte Argument

- Das erste Argument ist, im Fehlerfall, der Error.
- Der Callback wirft keine «Exception».

- Fast alle async-Methoden haben auch ein Synchrone Variante. Z.B. `readFileSync`.

- Diese wirft eine Exception bei einem Fehler

- Das Request und das Response Objekt ist ein Stream
- Streams sind «chainable».
- Gemeinsames Interface für alle “Streaming” Dienste
  - Netzwerk
  - File
- Unterschiedliche Varianten von Streams:
  - readable, writable, transform, duplex, "classic".

Weitere Informationen: <https://nodejs.org/api/stream.html>  
<https://github.com/substack/stream-handbook#introduction>



# Streams

```
/*Ohne Streams*/  
let server = http.createServer(function (req, res) {  
    fs.readFile(__dirname + '/data.txt', function (err, data) {  
        res.end(data);  
    });  
});
```

```
/*Mit Streams*/  
let server = http.createServer(function (req, res) {  
    let stream = fs.createReadStream(__dirname + '/data.txt');  
    stream.pipe(res);  
});
```

**Welcher dieser Varianten wäre vorzuziehen, wieso?**

- Node hat ein Module «event», welches ein «EventEmitter» beinhaltet

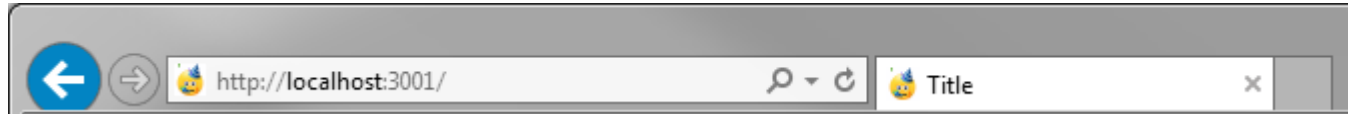
- <https://nodejs.org/api/events.html>

- Klassen können vom EventEmitter erben um dessen Eigenschaften zu übernehmen

```
const EventEmitter = require('events').EventEmitter;

class Door extends EventEmitter
{
  constructor() {
    super();
  }
  ring() {
    setTimeout(() => {
      this.emit('open');
    }, 100);
  };
}
```

**FAVICON**



- Kurz für «favorite icon»
  - shortcut icon, website icon, tab icon, URL icon oder bookmark icon.
- **Grösse**
  - Quadratisch: 16x16
  - Funktionieren in den meisten Browsern: 32x32, 64x64
- **Format**
  - \*.ico, \*.png, \*.jpg, \*.gif (nicht animiert)

## ■ Einbinden mit dem Link-Tag

```
<link rel="shortcut icon" type="image/gif" href="facepalm.gif" />
```

## ■ Oder das File auf dem Server am richtigen Ort platzieren



## ■ Oder beide Varianten kombiniert

- Rückwärtskompatibilität + Vorteil von Variante 1

**ÜBUNG**

# Übung 1

- Erstellen Sie ein Module, welches die Zahlen [VON, BIS] ausgibt.  
Der Start und End-Wert können dem Module übergeben werden.
  - Importieren Sie das Module in ein anders und rufen Sie dieses auf.
  - Optional: Versuchen Sie das Problem ohne Schleifen (while, for...) zu lösen

**Zeit: ca. 15 Minuten**

**Lösung:** [https://github.com/gfeller/CAS\\_FEE\\_NodeJS/tree/master/Uebungen/U1](https://github.com/gfeller/CAS_FEE_NodeJS/tree/master/Uebungen/U1)

## ■ Erstellen Sie ein Module, wessen API ein Filename und einen Text erwartet.

- Erstellen Sie dieses File mit dem übergebenem Text.
- Lesen Sie danach das File.
- Löschen Sie danach das File und geben Sie den gelesen Wert wieder zurück.
- Behandeln Sie auch den Fehlerfall.
- Nutzen Sie die API: <https://nodejs.org/api/fs.html>
- Optional: mit Async / Await

## ■ Schreiben Sie einen Node-Server, welcher mit 3 Urls umgehen kann.

- /number soll die Zahlen 0.....40 zum Client senden.
  - MIN und MAX aus der URL auslesen z.B. /number?min=10&max=20
- /file soll ein File erstellen, lesen und wieder löschen. Der Inhalt soll dem Client zurückgegeben werden.
- /sendFile soll ein HTML-File zum Client senden.
  - Das HTML File soll ein Formular beinhaltet. Diese Daten werden an Server geschickt und angezeigt.
- Bei allen anderen Anfragen soll ein 404er zurück gesendet werden

**Zeit: 30 Minuten**

**Lösung:** <https://github.com/gfeller/NodeJS/tree/master/Uebungen/U2>  
[https://github.com/gfeller/CAS\\_FEE\\_NodeJS/tree/master/Uebungen/U3](https://github.com/gfeller/CAS_FEE_NodeJS/tree/master/Uebungen/U3)